

# A note on approximate strengths of edges in a hypergraph

Chandra Chekuri\*

Chao Xu†

March 14, 2017

## Abstract

Let  $H = (V, E)$  be an edge-weighted hypergraph of rank  $r$ . Kogan and Krauthgamer [7] extended Benczúr and Karger’s [2] random sampling scheme for cut sparsification from graphs to hypergraphs. The sampling requires an algorithm for computing the approximate strengths of edges. In this note we extend the algorithm for graphs from [2] to hypergraphs and describe a near-linear time algorithm to compute approximate strengths of edges; we build on a sparsification result for hypergraphs from our recent work [4]. Combined with prior results we obtain faster algorithms for finding  $(1 + \varepsilon)$ -approximate mincuts when the rank of the hypergraph is small.

## 1 Introduction

Benczúr and Karger, in their seminal work [2], showed that all cuts of a weighted graph  $G = (V, E)$  on  $n$  vertices can be approximated to within a  $(1 \pm \varepsilon)$ -factor by a sparse weighted graph  $G' = (V, E')$  where  $|E'| = O(n \log n / \varepsilon^2)$ . Moreover,  $G'$  can be computed in near-linear time by a randomized algorithm. The algorithm has two steps. In the first step, it computes for each edge  $e$  a number  $p_e$  which is proportional to the inverse of the approximate strength of edge  $e$ . The second step is a simple sampling scheme where each edge is independently sampled with probability  $p_e$  and if it is chosen, the edge  $e$  is assigned a capacity  $u_e / p_e$  where  $u_e$  is the original capacity/weight of  $e \in E$ . It is shown in [2] that the probabilities  $p_e, e \in E$  can be computed by a deterministic algorithm in  $O(m \log^3 n)$  time where  $m = |E|$  if  $G$  is weighted and in  $O(m \log^2 n)$  time if  $G$  is unweighted or has polynomially bounded weights. More recent work [5] has shown a general framework for cut sparsification where the sampling probability  $p_e$  can also be chosen to be approximate connectivity between the end points of  $e$ . In another seminal work, Batson, Spielman and Srivastava [1] showed that spectral-sparsifiers, which are stronger than cut-sparsifiers, with  $O(n/\varepsilon^2)$  edges exist, and can be computed in polynomial time. Lee and Sun recently showed such sparsifiers can be computed in  $\tilde{O}(m)$  time, where  $\tilde{O}$  hide polylog factors [10].

In this paper, we are interested in hypergraphs.  $H = (V, E)$  is a hypergraph where each  $e \in E$  is a subset of nodes. Graphs are a special case when each  $e$  has cardinality 2. The rank of a hypergraph  $H = (V, E)$  is  $\max_{e \in E} |e|$ . Recently Kogan and Krauthgamer [7] extended Benczúr and Karger’s sampling scheme and analysis to show the following: for any weighted hypergraph  $H = (V, E)$  with rank  $r$  there is a  $(1 \pm \varepsilon)$ -approximate cut-sparsifier with  $O(n(r + \log n)/\varepsilon^2)$  edges. They show that sampling with (approximate) strengths will yield the desired sparsifier. Finding the strengths of edges in hypergraphs can be easily done in polynomial time. In this note, we develop a near-linear time algorithm for computing approximate strengths of edges in a weighted hypergraph. We state a formal theorem and indicate some applications after we formally define strength of an edge.

\*Department of Computer Science, University of Illinois, Urbana, IL 61801. CHEKURI@ILLINOIS.EDU. Work on this paper supported in part by NSF grant CCF-1526799.

†Department of Computer Science, University of Illinois, Urbana, IL 61801. CHAOXU3@ILLINOIS.EDU. Work on this paper supported in part by NSF grant CCF-1526799.

**Strength of an edge:** Let  $H = (V, E)$  be an *unweighted* hypergraph. For  $U \subseteq V$ ,  $H[U] = (U, \{e \mid e \subseteq U, e \in E\})$  denotes the vertex induced subhypergraph of  $H$ . For  $A \subseteq V$  we denote by  $\delta_H(A)$  the set of edges that cross  $A$ ; formally  $\delta_H(A) = \{e \in E \mid e \cap A \neq \emptyset, e \cap (V \setminus A) \neq \emptyset\}$ . A hypergraph is  $k$ -edge-connected if  $|\delta_H(A)| \geq k$  for every non-trivial cut  $\emptyset \subsetneq A \subsetneq V$ . The **edge-connectivity** of  $H$ , denoted by  $\lambda(H)$ , is the largest  $k$  such that  $H$  is  $k$ -edge-connected. Equivalently,  $\lambda(H)$  is the value of the min-cut in  $H$ . The **strength** of an edge  $e$ , denoted by  $\gamma_H(e)$ , is defined as  $\max_{U \subseteq V} \lambda(H[U])$ ; in other words the largest connectivity of a vertex induced subhypergraph that contains  $e$ . We also define the cost  $\zeta_H(e)$  of  $e$  as the inverse of the strength; that is  $\zeta_H(e) = 1/\gamma_H(e)$ . We drop the subscript  $H$  if there is no confusion regarding the hypergraph.

The preceding definitions generalize easily to weighted hypergraphs. Let  $w(e)$  be a non-negative integer weight for edge  $e$ . Then the notion of edge-connectivity easily generalizes via the weight of a mincut. The strength of an edge  $e$  is the maximum mincut value over all vertex induced subhypergraphs that contain  $e$ .

For a hypergraph  $H$  we use  $n$  to denote the number of vertices,  $m$  to denote the number of edges and  $p = \sum_{e \in E(H)} |e|$  to denote the total degree. We observe that  $p$  is the natural representation size of  $H$  when the rank of the hypergraph is not bounded.

Our main technical result is the following.

**Theorem 1.1** *Let  $H = (V, E)$  be a edge-weighted hypergraph on  $n$  vertices. There is an efficient algorithm that computes for each edge  $e$  an approximate strength  $\gamma'(e)$  such that the following properties are satisfied:*

1. *lower bound property:*  $\gamma'(e) \leq \gamma_H(e)$  and
2. *c-cost property:*  $\sum_{e \in E} \frac{1}{\gamma'(e)} \leq c(n-1)$  where  $c = O(r)$ .

*For unweighted hypergraphs the running time of the algorithm is  $O(p \log^2 n)$  and for weighted hypergraphs the running time is  $O(p \log p \log^2 n)$ .*

The function that satisfies the theorem is called a  $c$ -approximate strength. One natural approach is converting the hypergraph to a graph, and hope the strength approximately carries over. For example, replacing each hyperedge with a clique, or a star spanning the vertices in the edge. The minimum strength of the replaced edges might give us a  $O(c)$ -approximation. Unfortunately, this will not work even when rank is only 3. Consider the following hypergraph  $H$  with vertices  $\{v_1, \dots, v_n\}$ . There is an edge  $e = \{v_1, v_2\}$ , and edge  $\{v_1, v_2, v_i\}$  for all  $3 \leq i \leq n$ . The strength of  $e$  in  $H$  is 1. Let  $G$  be a graph where each  $\{v_1, v_2, v_i\}$  in  $H$  is replaced with a star centered at  $v_1$  and spans  $\{v_1, v_2, v_i\}$ . The strength of  $e$  in  $G$  is  $n-1$ . This bound also holds if each hyperedge  $\{v_1, v_2, v_i\}$  is replaced by a clique.

Our proof of the preceding theorem closely follows the corresponding theorem for graphs from [2]. A key technical tool for graphs is the deterministic sparsification algorithm for edge-connectivity due to Nagamochi and Ibaraki [12]. Here we rely on a generalization to hypergraphs from our recent work [4].

## 1.1 Applications

A weighted hypergraph  $H' = (V, E')$  is a  $(1 \pm \varepsilon)$ -cut approximation of a weighted hypergraph  $H = (V, E)$ , if for every  $S \subseteq V$ , the cut value of  $S$  in  $H'$  is within  $(1 \pm \varepsilon)$  factor of the cut value of  $S$  in  $H$ . Below we state formally the sampling theorem from [7].

**Theorem 1.2** ([7]) *Let  $H$  be a rank  $r$  weighted hypergraph where edge  $e$  has weight  $w(e)$ . Let  $H_\varepsilon$  be a hypergraph obtained by independently sampling each edge  $e$  in  $H$  with probability  $p_e = \min\left(\frac{3((d+2)\ln n + r)}{\gamma_H(e)\varepsilon^2}, 1\right)$  and weight  $w(e)/p_e$  if sampled. With probability at least  $1 - O(n^{-d})$ , the hypergraph  $H_\varepsilon$  has  $O(n(r + \log n)/\varepsilon^2)$  edges and is a  $(1 \pm \varepsilon)$ -cut-approximation of  $H$ .*

The expected weight of each edge in  $H_\varepsilon$  is the weight of the edge in  $H$ . It is not difficult to prove that  $c$ -approximate strength also suffices to get a  $(1 \pm \varepsilon)$ -cut-approximation. That is, if we replace  $\gamma$  with  $c$ -approximate strength  $\gamma'$ , the sampling algorithm will still output a  $(1 \pm \varepsilon)$ -cut-approximation of  $H$ . Indeed, the lower bound property shows each edge will be sampled with a higher probability, therefore the probability of obtaining a  $(1 \pm \varepsilon)$ -sparsifier increases. However, the number of edges in the sparsifier increases. The  $c$ -cost property shows the hypergraph  $H_\varepsilon$  will have  $O(cn(r + \log n)/\varepsilon^2)$  edges.

From [Theorem 1.1](#), we can find an  $O(r)$ -approximate strength function  $\gamma'$  in  $O(p \log^2 n \log p)$  time.

**Corollary 1.3** *A  $(1 \pm \varepsilon)$ -cut approximation of  $H$  with  $O(nr(r + \log n)/\varepsilon^2)$  edges can be found in  $O(p \log^2 n \log p)$  time with high probability.*

The number of edges in the  $(1 \pm \varepsilon)$ -cut approximation is worse than [Theorem 1.2](#) by a factor of  $r$ . It is an open problem whether the extra factor of  $r$  can be removed.

As is the case for graphs, cut sparsification allows for faster approximation algorithms for cut problems. We mention two below.

**Mincut:** The best running time known currently to compute a (global) mincut in a hypergraph is  $O(pn)$  [6, 11]. A  $(2 + \varepsilon)$ -approximation can be computed in  $\tilde{O}(p/\varepsilon)$  time [4]. Via [Corollary 1.3](#) we can first sparsify the hypergraph and then apply the  $O(pn)$  time algorithm to the sparsified graph. This gives a randomized algorithm that outputs a  $(1 + \varepsilon)$ -approximate mincut in a rank  $r$  hypergraph in  $O(n^2 r^2 (r + \log n)/\varepsilon^2 + p \log^2 n \log p)$  time and works with high probability. For small  $r$  the running time is  $\tilde{O}(p + n^2/\varepsilon^2)$  for a  $(1 + \varepsilon)$ -approximation.

**$s$ - $t$  mincut:** The standard technique to compute a  $s$ - $t$  mincut in a hypergraph is computing a  $s$ - $t$  maximum flow in an associated capacitated digraph with  $O(n + m)$  vertices and  $O(p)$  edges [8]. A  $(1 - \varepsilon)$  approximation algorithm of  $s$ - $t$  maximum flow in such graph can be found in  $\tilde{O}(p\sqrt{n + m})$  time [9]. Via sparsification [Corollary 1.3](#), we can obtain a randomized algorithm to find a  $(1 + \varepsilon)$ -approximate  $s$ - $t$  mincut in a rank  $r$  hypergraph in  $\tilde{O}(n^{3/2} r^2 (r + \log n)^{3/2}/\varepsilon^3 + p)$  time. For small  $r$  the running time is  $\tilde{O}(p + n^{3/2}/\varepsilon^3)$  for a  $(1 + \varepsilon)$ -approximation.

## 2 Preliminaries

A  **$k$ -strong component** of  $H$  is a inclusion-wise maximal set of vertices  $U \subseteq V$  such that  $H[U]$  is  $k$ -edge-connected. An edge  $e$  is  **$k$ -strong** if  $\gamma(e) \geq k$ , otherwise  $e$  is  **$k$ -weak**.  $\kappa(H)$  is the number of components of a hypergraph  $H$ . The **size** of  $H$  is  $\sum_{e \in E} |e|$ . The **degree** of a vertex  $v$ ,  $\deg(v)$ , is the number of edges incident to  $v$ .  $n$  denotes the number of vertices in  $H$ ,  $p$  denotes the size of  $H$  and  $r$  denotes the rank of  $H$ .

Contracting an edge  $e$  in a hypergraph  $H = (V, E)$  results in a new hypergraph  $H' = (V', E')$  where all vertices in  $e$  are identified into a single new vertex  $v_e$ . Any edge  $e' \subseteq e$  is removed. Any edge  $e'$  that properly intersects with  $e$  is adjusted by replacing  $e' \cap e$  by the new vertex  $v_e$ . We note  $H'$  by  $H/e$ .

**Proposition 2.1** *Deleting an edge does not increase the strength of any remaining edge. Contracting an edge does not decrease the strength of any remaining edge.*

**Lemma 2.2** *Let  $H = (V, E)$  be a hypergraph. If an edge  $e \in E$  crosses a cut of value  $k$ , then  $\gamma_H(e) \leq k$ .*

**Proof:** Suppose  $S$  is a cut such that  $|\delta_H(S)| \leq k$  and  $e \in \delta_H(S)$ . Consider any  $U \subseteq V$  that contains  $e$  as a subset and let  $H' = H[U]$ . It is easy to see that  $|\delta_{H'}(S \cap U)| \leq k$  and hence  $\lambda(H') \leq k$ . Therefore,  $\gamma(e) = \max_{e \subseteq U \subseteq V} \lambda(H[U]) \leq k$ .  $\square$

**Lemma 2.3 (Extends Lemma 4.5, 4.6 [2])** *Let  $e, e'$  be distinct edges in a hypergraph  $H = (V, E)$ .*

1. *If  $\gamma_H(e) \geq k$  and  $e'$  is a  $k$ -weak edge then  $\gamma_H(e) = \gamma_{H'}(e)$  where  $H' = H \setminus e'$ .*

2. Suppose  $\gamma_H(e) < k$  (that is  $e$  is a  $k$ -weak edge) and  $e'$  is a  $k$ -strong edge. Let  $H' = H/e'$  obtained by contracting  $e'$ . If  $f$  is the corresponding edge of  $e$  in  $H'$  then  $\gamma_H(e) = \gamma_{H'}(f)$ .

In short, contracting a  $k$ -strong edge does not increase the strength of a  $k$ -weak edge and deleting a  $k$ -weak edge does not decrease the strength of a  $k$ -strong edge.

**Proof:** We prove the two claims separately.

**Deleting a  $k$ -weak edge:** Since  $e$  is  $k$ -strong in  $H$  there is  $U \subseteq V$  such that  $\lambda(H[U]) \geq k$  and  $e \subseteq U$ . If  $e' \subseteq U$ ,  $e'$  would be a  $k$ -strong edge. Since  $e'$  is  $k$ -weak,  $e'$  does not belong to  $H[U]$ . Thus  $H'[U] = H[U]$  which implies that  $\gamma_{H'}(e) \geq k$ .

**Contracting a  $k$ -strong edge:** Let  $H' = H/e'$  where  $e'$  is a  $k$ -strong edge in  $H$ . Let  $v_{e'}$  be the vertex in  $H'$  obtained by contracting  $e'$ . Let  $U' \subseteq V(H')$  be the set that certifies the strength of  $f$ , that is  $\gamma_{H'}(f) = \lambda(H'[U'])$  and  $f \subseteq U'$ . If  $v_{e'} \notin U'$  then it is easy to see that  $U' \subseteq V(H)$  and  $H[U] = H'[U']$  which would imply that  $\gamma_{H'}(f) = \gamma_H(e)$ . Thus, we can assume that  $v_{e'} \in U'$ . Let  $U$  be the set of vertices in  $V(H)$  obtained by uncontracting  $v_{e'}$ . We claim that  $\lambda(H[U]) \geq \lambda(H'[U'])$ . If this is the case we would have  $\gamma_H(e) \geq \lambda(H[U]) \geq \lambda(H'[U']) = \gamma_{H'}(f)$ . We now prove the claim. Let  $W$  be the set that certifies the strength of  $e'$ , namely  $\gamma_H(e') = \lambda(H[W])$ . Consider any cut  $S \subseteq U \cup W$  in  $H[U \cup W]$ . If  $S$  crosses  $W$  or strictly contained in  $W$ , then  $S$  has cut value at least  $\lambda(H[W])$ . Otherwise,  $S \subseteq U \setminus W$  or  $W \subseteq S$ . By symmetry, we only consider  $S \subseteq U \setminus W$ .  $S \neq U$  because  $e' \subseteq U \cap W$ .  $S$  is also a cut in  $H'[U']$ , and  $|\delta_{H[U]}(S)| = |\delta_{H'[U']}(S)|$ . This shows that

$$\lambda(H[U \cup W]) \geq \min(\lambda(H'[U']), \lambda(H[W]))$$

Because  $\lambda(H[W]) > \lambda(H[U])$ , and  $\lambda(H[U]) \geq \lambda(H[U \cup W])$ , we arrive at  $\lambda(H[U]) \geq \lambda(H'[U'])$ .  $\square$

**Theorem 2.4 (Extends Lemma 4.10 [2])** Consider a connected unweighted hypergraph  $H = (V, E)$ . A weighted hypergraph  $H' = (V, E)$  with weight  $\zeta_H(e)$  on each edge  $e$  has minimum cut value 1.

**Proof:** Consider a cut  $S$  of value  $k$  in  $H$ ; that is  $|\delta_H(S)| = k$ . For each edge  $e \in \delta(S)$ ,  $\gamma_H(e) \leq k$  by Lemma 2.2. Therefore  $e$  has weight at least  $1/k$  in  $H'$ . It follows the cut value of  $S$  in  $H'$  is at least  $k \cdot 1/k \geq 1$ . Thus, the mincut of  $H'$  is at least 1.

Let  $S$  be a mincut in  $H$  whose value is  $k^*$ . It is easy to see that for each  $e \in \delta_H(S)$ ,  $\gamma_H(e) = k^*$ . Thus the value of the cut  $S$  in  $H'$  is exactly  $k^* \cdot 1/k^* = 1$ .  $\square$

**Lemma 2.5 (Extends Lemma 4.11 [2])** For a unweighted hypergraph  $H = (V, E)$  with at least 1 vertex,

$$\sum_{e \in E} \zeta(e) \leq n - \kappa(H)$$

**Proof:** Let  $t = n - \kappa(H)$ . We prove the theorem by induction on  $t$ .

For the base case, if  $t = 0$ , then  $H$  has no edges and therefore the sum is 0.

Otherwise, let  $t > 0$ . Let  $U$  be a connected component of  $H$  with at least 2 vertices. There exists a cut  $X$  of cost 1 in  $H[U]$  by Theorem 2.4.

Let  $H' = H - \delta(S)$ .  $H'$  has at least one more connected component than  $H$ . By Proposition 2.1, for each  $e \in E(H')$   $\gamma_H(e) \geq \gamma_{H'}(e)$ ; hence  $\zeta_{H'}(e) \geq \zeta_H(e)$ . By the inductive hypothesis,  $\sum_{e \in E(H')} \zeta_{H'}(e) \leq (n - \kappa(H')) \leq t - 1$ . The edges in  $H$  are exactly  $\delta(X) \cup E(H')$ . By the same argument as in the preceding

```

ESTIMATION( $H$ )
  Compute a number  $k$  such that  $\gamma_H(e) \geq k$  for all  $e \in E(H)$ 
   $H_0 \leftarrow H, i \leftarrow 1$ 
  while there are edges in  $H_{i-1}$ 
     $F_i \leftarrow \text{WEAKEDGES}(H_{i-1}, 2^i k)$ 
    for  $e \in F_i$ 
       $\gamma'(e) \leftarrow 2^{i-1} k$ 
     $H_i \leftarrow H_{i-1} - F_i$ 
     $i \leftarrow i + 1$ 
  return  $\gamma'$ 

```

Figure 3.1: The estimation algorithm

lemma,  $\sum_{e \in \delta(X)} \zeta_H(e) = 1$ . Therefore,

$$\begin{aligned}
\sum_{e \in E(H)} \zeta_H(e) &= \sum_{e \in \delta(X)} \zeta_H(e) + \sum_{e \in E(H')} \zeta_H(e) \\
&= 1 + \sum_{e \in E(H')} \zeta_H(e) \\
&\leq 1 + \sum_{e \in E(H')} \zeta_{H'}(e) \\
&\leq 1 + (t - 1) = t.
\end{aligned}$$

□

**Corollary 2.6** *The number of  $k$ -weak edges in an unweighted hypergraph  $H$  on  $n$  vertices is at most  $k(n - \kappa(H))$ .*

**Lemma 2.7** *The  $k$ -strong components are pairwise disjoint.*

**Proof:** Consider  $k$ -strong components  $A$  and  $B$ . Assume  $A \cap B \neq \emptyset$ , then  $\lambda(G[A \cup B]) \geq \min(\lambda(G[A]), \lambda(G[B])) \geq k$  using triangle inequality for connectivity. This shows  $A = A \cup B = B$  by maximality of  $A$  and  $B$ . □

### 3 Estimating strengths in unweighted hypergraphs

In this section, we consider unweighted hypergraphs and describe a near-linear time algorithm to estimate the strengths of all edges as stated in [Theorem 1.1](#). Let  $H = (V, E)$  be the given hypergraph. The high-level idea is simple. We assume that there is a fast algorithm  $\text{WEAKEDGES}(H, k)$  that returns a set of edges  $E' \subseteq E$  such that  $E'$  contains all the  $k$ -weak edge in  $E$ ; the important aspect here is that the output may contain some edges which are *not*  $k$ -weak, however, the algorithm should not output too many such edges (this will be quantified later).

The estimation algorithm is defined in [Figure 3.1](#). The algorithm repeatedly calls  $\text{WEAKEDGES}(H, k)$  for increasing values of  $k$  while removing the edges found in previous iterations.

**Lemma 3.1** *Let  $H = (V, E)$  be an unweighted hypergraph. Then,*

1. *For each  $e \in F_i$ ,  $\gamma(e) \geq 2^{i-1}k$ . That is, the strength of all edges deleted in iteration  $i$  is at least  $2^{i-1}k$ .*
2. *For each  $e \in E$ ,  $\gamma'(e) \leq \gamma(e)$ .*

**Proof:**  $\gamma_{H_i}(e) \leq \gamma_H(e)$  for all  $i$  and  $e \in E(H_i)$  because deleting edges cannot increase strength. Let  $E_i$  denote the set of edges in  $H_i$  with  $E_0 = E$ .

We prove that  $2^i k \leq \gamma_{H_i}(e)$  for all  $e \in E_i$  by induction on  $i$ . If  $i = 0$ , then  $k \leq \gamma_{H_0}(e)$  for all  $e \in E_0$ . Now we assume  $i > 0$ . At end of iteration  $(i - 1)$ , by induction, we have that  $\gamma_{H_{i-1}}(e) \geq 2^{i-1} k$  for each  $e \in E_{i-1}$ . In iteration  $i$ ,  $F_i$  contains all  $2^i k$ -weak edges in the graph  $H_{i-1}$ . We have  $E_i = E_{i-1} - F_i$ . Thus, for any edge  $e \in E_i$ ,  $\gamma_H(e) \geq \gamma_{H_{i-1}}(e) \geq 2^i k$ . This proves the claim.

Since  $F_i \subseteq E_{i-1}$ , it follows from the previous claim that  $\gamma_H(e) \geq 2^{i-1} k$  for all  $e \in F_i$ . Since the  $F_i$  form a partition of  $E$ , we have  $\gamma'(e) \leq \gamma_H(e)$  for all  $e \in E$ .  $\square$

Note that in principle  $\text{WEAKEDGES}(H, k)$  could output all the edges of the graph for any  $k \geq \lambda(H)$ . This would result in a high cost for the resulting strength estimate. Thus, we need some additional properties on the output of the procedure  $\text{WEAKEDGES}(H, k)$ . Let  $H = (V, E)$  be a hypergraph. A set of edges  $E' \subseteq E$  is called  $\ell$ -**light** if  $|E'| \leq \ell(\kappa(H - E') - \kappa(H))$ . Intuitively, on average, we remove  $\ell$  edges in  $E'$  to increase the number of components of  $H$  by 1.

**Lemma 3.2** *If  $\text{WEAKEDGES}(H, k)$  outputs a set of  $ck$ -light edges for all  $k$ , then the output  $\gamma'$  of the algorithm  $\text{ESTIMATION}(H)$  satisfies the  $2c$ -cost property. That is,  $\sum_{e \in E} \frac{1}{\gamma'(e)} \leq 2c(n - 1)$ .*

**Proof:** From the description of  $\text{ESTIMATION}(H)$ , and using the fact that the edges sets  $F_1, F_2, \dots$ , partition  $E$ , we have  $\sum_{e \in E} \frac{1}{\gamma'(e)} = \sum_{i \geq 1} |F_i| \frac{1}{2^{i-1} k}$ .

$F_i$  is the output of  $\text{WEAKEDGES}(H_{i-1}, 2^i k)$ . From the lightness property we assumed,  $|F_i| \leq c 2^i k (\kappa(H_i) - \kappa(H_{i-1}))$ . Combining this with the preceding equality,

$$\sum_{e \in E} \frac{1}{\gamma'(e)} = \sum_{i \geq 1} |F_i| \frac{1}{2^{i-1} k} \leq \sum_{i \geq 1} 2c(\kappa(H_i) - \kappa(H_{i-1})) \leq 2c(n - 1).$$

$\square$

### 3.1 Implementing WEAKEDGES

We now show describe an implementation of  $\text{WEAKEDGES}(H, k)$  that outputs a  $4rk$ -light set.

Let  $H = (V, E)$  be a hypergraph. An edge  $e$  is  **$k$ -crisp** with respect to  $H$  if it crosses a cut of value less than  $k$ . In other words, there is a cut  $X$ , such that  $e \in \delta(X)$  and  $|\delta(X)| < k$ . Note that any  $k$ -crisp edge is  $k$ -weak. A set of edges  $E' \subseteq E$  is a  **$k$ -partition**, if  $E'$  contains all the  $k$ -crisp edges in  $H$ . A  $k$ -partition may contain non- $k$ -crisp edges.

We will assume access to a subroutine  $\text{PARTITION}(H, k)$  that given  $H$  and integer  $k$ , it finds a  $2k$ -light  $k$ -partition of  $H$ . We will show how to implement  $\text{PARTITION}$  later. See Figure 3.2 for the implementation of  $\text{WEAKEDGES}$ .

```

WEAKEDGES( $H, k$ )
 $E' \leftarrow \emptyset$ 
repeat  $1 + \log_2 n$  times:
     $E' \leftarrow E' \cup \text{PARTITION}(H, 2rk)$ 
     $H \leftarrow H - E'$ 
return  $E'$ 

```

Figure 3.2: Algorithm for returning a  $4rk$ -light set of all  $k$ -weak edges in  $H$ .

**Theorem 3.3**  $\text{WEAKEDGES}(H, k)$  returns a  $4rk$ -light set  $E'$  such that  $E'$  contains all the  $k$ -weak edges of  $H$  with  $O(\log n)$  calls to  $\text{PARTITION}$ .



```

PARTITION( $H, k$ )
  if number of edges in  $H \leq 2k(n - \kappa(H))$ 
     $E' \leftarrow$  edges in  $H$ 
    return  $E'$ 
  else
     $E' \leftarrow$  CERTIFICATE( $H, k$ )
     $H \leftarrow$  contract all edges of  $H - E'$ 
    return PARTITION( $H, k$ )

```

Figure 3.3: Algorithm for returning a  $2k$ -light  $k$ -partition.

**Proof:** First, we assume  $H$  has no  $k$ -strong component with more than 1 vertex and that  $H$  is connected. Then all edges are  $k$ -weak and the number of  $k$ -weak edges in  $H$  is at most  $k(n - 1)$  by [Corollary 2.6](#). It also implies that  $\sum_v \deg(v) \leq rk(n - 1)$ . By Markov's inequality at least half the vertices have degree less than  $2rk$ . For any vertex  $v$  with degree less than  $2rk$ , all edges incident to it are  $2rk$ -crisp. Thus, after the first iteration, all such vertices become isolated since  $\text{PARTITION}(H, 2rk)$  contains all  $2rk$ -crisp edges. If  $H$  is not connected then we can apply this same argument to each connected component and deduce that at least half of the vertices in each component will be isolated in the first iteration. Therefore, in  $\log n$  iterations, all vertices become isolated. Hence  $\text{WEAKEDGES}(H, k)$  returns all the edges of  $H$ .

Now consider the general case when  $H$  may have  $k$ -strong components. We can apply the same argument as above to the hypergraph obtained by contracting each  $k$ -strong component into a single vertex. This is well defined because the  $k$ -strong components are disjoint by [Lemma 2.7](#).

Let the edges removed in the  $i$ th iteration be  $E_i$ , and the hypergraph before the edge removal be  $H_i$ . So  $H_{i+1} = H_i - E_i$ . Recall that  $\text{PARTITION}(H_i, 2rk)$  returns a  $4rk$ -light set. Hence we know  $|E_i| \leq 4rk(\kappa(H_{i+1}) - \kappa(H_i))$ .

$$|E'| = \sum_{i \geq 1} |E_i| \leq \sum_{i \geq 1} 4rk(\kappa(H_{i+1}) - \kappa(H_i)) = 4rk(\kappa(H - E') - \kappa(H))$$

This shows  $E'$  is  $4rk$ -light. □

It remains to implement  $\text{PARTITION}(H, k)$  that returns a  $2k$ -light  $k$ -partition. To do this, we introduce  $k$ -sparse certificates. Let  $H = (V, E)$  be a hypergraph. A subset of edges  $E' \subseteq E$ , define  $H' = (V, E')$  is a  **$k$ -sparse certificate** if  $|\delta_{H'}(A)| \geq \min(|\delta_H(A)|, k)$  for all  $A \subseteq V$  where  $H' = (V, E')$ .

**Theorem 3.4** *There is an algorithm  $\text{CERTIFICATE}(H, k)$  that given hypergraph  $H$  and integer  $k$ , finds a  $k$ -sparse certificate  $E'$  of  $H$  in  $O(p)$  time such that  $|E'| \leq k(n - 1)$ .*

**Proof:** See [Appendix A](#). □

A  $k$ -sparse certificate  $E'$  is certainly a  $k$ -partition. However,  $E'$  may contain too many edges to be  $2k$ -light. Thus, we would like to find a smaller subset of  $E'$ . Note that every  $k$ -crisp edge must be in a  $k$ -sparse certificate and hence no edge in  $E \setminus E'$  can be  $k$ -crisp. Hence we will contract the edges in  $E \setminus E'$ , and find a  $k$ -sparse certificate in the hypergraph after the contraction. We repeat the process until eventually we reach a  $2k$ -light set. See [Figure 3.3](#) for the formal description of the algorithm.

**Theorem 3.5**  *$\text{PARTITION}(H, k)$  outputs a  $2k$ -light  $k$ -partition in  $O(p \log n)$  time.*

**Proof:** If the algorithm either returns all the edges of the graph  $H$  in the first step then it is easy to see that the output is a  $2k$ -light  $k$ -partition since the algorithm explicitly checks for the lightness condition.

Otherwise let  $E'$  be the output of  $\text{CERTIFICATE}(H, k)$ . As we argued earlier,  $E - E'$  contains no  $k$ -crisp edges and hence contracting them is safe. Moreover, all the original  $k$ -crisp edges remain  $k$ -crisp after the contraction. Since the algorithm recurses on the new graph, this establishes the correctness of the output.

We now argue for termination and running time by showing that the number of vertices halves in each recursive call. Assume  $H$  contains  $n$  vertices, the algorithm finds a  $k$ -sparse certificate and contracts all edges not in the certificate. The resulting hypergraph has  $n'$  vertices and  $m'$  edges. We have  $m' \leq k(n-1)$  by [Theorem 3.4](#). If  $n' - 1 \leq (n-1)/2$ , then the number of vertices halved. Otherwise  $n' - 1 > (n-1)/2$ , then the number of edges  $m' \leq k(n-1) < 2k(n' - 1)$ , and the algorithm terminates in the next recursive call.

The running time of the algorithm for a size  $p$  hypergraph with  $n$  vertices is  $T(p, n)$ .  $T(p, n)$  satisfies the recurrence  $T(p, n) = O(p) + T(p, n/2) = O(p \log n)$ .  $\square$

Putting things together,  $\text{ESTIMATION}(H)$  finds the desired  $O(r)$ -approximate strength.

**Theorem 3.6** *Let  $H = (V, E)$  be a unweighted hypergraph. The output  $\gamma'$  of  $\text{ESTIMATION}(H)$  is a  $O(r)$ -approximate strength function of  $H$ .  $\text{ESTIMATION}(H)$  can be implemented in  $O(p \log^2 n \log p)$  time.*

**Proof:** Combining [Lemma 3.1](#), [Lemma 3.2](#) and [Theorem 3.3](#), we get the output  $\gamma'$  of  $\text{ESTIMATION}(H)$  is a  $O(r)$ -approximate strength function. The maximum strength in the graph is at most  $p$ , all edges with be removed at the  $(1 + \log p)$ th iteration of the while loop. In each iteration, there is one call to  $\text{WEAKEDGES}$ . Each call of  $\text{WEAKEDGES}$  takes  $O(p \log^2 n)$  time by combining [Theorem 3.5](#) and [Theorem 3.3](#). The step outside the while loop takes linear time, since we can set  $k$  to be 1 as a lower bound of the strength. Hence overall, the running time is  $O(p \log^2 n \log p)$ .  $\square$

## 4 Estimating strengths in weighted hypergraphs

Consider a weighted hypergraph  $H = (V, E)$  with an associated weight function  $w : E \rightarrow \mathbb{N}_+$ ; that is, we assume all weights are non-negative integers. For proving correctness, we consider an unweighted hypergraph  $H'$  that simulates  $H$ . Let  $H'$  contain  $w(e)$  copies of edge  $e$  for every edge  $e$  in  $H$ ; one can see that the strength of each of the copies of  $e$  in  $H'$  is the same as the strength of  $e$  in  $H$ . Thus, it suffices to compute strengths of edges in  $H'$ . We can apply the correctness proofs from the previous sections to  $H'$ . In the remainder of the section we will only be concerned with the running time issue since we do not wish to explicitly create  $H'$ . We say  $H$  is the implicit representation of  $H'$ .

We can implement  $\text{CERTIFICATE}(H, k)$  that finds an implicit representation of the  $k$ -sparse certificate  $E'$  of  $H'$  such that  $|E'| \leq k(n-1)$ , in  $O(p + n \log n)$  time, where  $p$  is the number of edges in the implicit representation, see [Appendix A](#).

The remaining operations in  $\text{PARTITION}$  and  $\text{WEAKEDGES}$  consist only of adding edges, deleting edges and contracting edges. These operations take time only depending on the size of the implicit representation. Therefore the running time in [Theorem 3.5](#) and [Theorem 3.3](#) still holds for weighted hypergraph.

**Theorem 4.1** *Given  $b$  a lower bound of the strength. If the total weight of all edges is at most  $bM$ , then  $\text{ESTIMATION}(H)$  can be implemented in  $O(p \log^2 n \log M)$  time.*

**Proof:** Because  $b$  is a lower bound of the strength, we can set  $k$  to be  $b$  in the first step. The maximum strength in the graph is at most  $bM$ , all edges will be removed at the  $(1 + \log M)$ th iteration of the while loop. Each iteration calls  $\text{WEAKEDGES}$  once, hence the running time is  $O(p \log^2 n \log M)$ .  $\square$



The running time in [Theorem 4.1](#) can be improved to strongly polynomial time by using the windowing technique [2].

Assume we have disjoint intervals  $I_1, \dots, I_t$  where for every  $e \in E$ ,  $\gamma(e) \in I_i$ . In addition, assume  $I_i = [a_i, b_i]$ ,  $b_i \leq p^2 a_i$  and  $b_i \leq a_{i+1}$  for all  $i$ . We can essentially apply the estimation algorithm to edges with strength inside each interval. Let  $E_i$  be the set of edges whose strength lies in interval  $I_i$ . Indeed, let  $H_i$  to be the graph obtained from  $H$  by contracting all edges in  $E_j$  where  $j > i$ , and deleting all edges in  $E_{j'}$  where  $j' < i$ . For edge  $e \in E_i$  let  $e'$  be its corresponding edge in  $H_i$ . From [Lemma 2.3](#),  $\gamma_{H_i}(e') = \gamma_H(e)$ . The total weight of  $H_i$  is at most  $p b_i \leq p^3 a_i$ . We can run  $\text{ESTIMATION}(H_i)$  to estimate  $\gamma_{H_i}$  since the ratio between the lower bound  $a_i$  and upper bound  $b_i$  is  $p^3$ . Let  $p_i$  be the size of  $H_i$ , and  $n_i$  be the number of vertices in  $H_i$ , the running time for  $\text{ESTIMATION}(H_i)$  is  $O(p_i \log^2 n_i \log p_i)$  by [Theorem 4.1](#). The total running time of  $\text{ESTIMATION}$  over all  $H_i$  is  $O(\sum_i p_i \log^2 n_i \log p_i) = O(p \log^2 n \log p)$ . Constructing  $H_i$  from  $H_{i+1}$  takes  $O(p_i + p_{i+1})$  time: contract all edges in  $H_{i+1}$  and then add all the edges  $e_i$  where  $\gamma(e) \in I_i$ . Therefore we can construct all  $H_1, \dots, H_t$  in  $O(p)$  time.

It remains to find the intervals  $I_1, \dots, I_t$ . For each edge  $e$ , we first find values  $d_e$ , such that  $d_e \leq \gamma(e) \leq p d_e$ . The maximal intervals in  $\bigcup_{e \in E} [d_e, p d_e]$  are the desired intervals. We now describe the procedure to find the values  $d_e$  for all  $e \in E$ .

**Definition** The star approximate graph  $A(H)$  of  $H$  is a weighted graph obtained by replacing each hyperedge  $e$  in  $H$  with a star  $S_e$ , where the center of the star is an arbitrary vertex in  $e$ , and the star spans each vertex in  $e$ . Every edge in  $S_e$  has weight equal to the weight of  $e$ .

It is important that  $A(H)$  is a multigraph: parallel edges are distinguished by which hyperedge it came from. We define a correspondence between the edges in  $A(H)$  and  $H$  by a function  $\pi$ . For an edge  $e'$  in  $A(H)$ ,  $\pi(e') = e$  if  $e' \in S_e$ . Let  $T$  be a maximum weight spanning tree in  $A(H)$ . For  $e \in E$ , define  $T_e$  to be the minimal subtree of  $T$  that contains all vertices in  $e$ . Note that all the leaves of  $T_e$  are vertices from  $e$ .

For any two vertices  $u$  and  $v$ , we define  $d_{uv}$  to be the weight of the minimum weight edge in the unique  $u$ - $v$ -path in  $T$ . For each edge  $e \in E$  we let  $d_e = \min_{u,v \in e} d_{uv}$ . We will show  $d_e$  satisfies the property that  $d_e \leq \gamma(e) \leq p d_e$ .

Let  $V_e = \bigcup_{e' \in T_e} \pi(e')$ . Certainly,  $\gamma(e) \geq \lambda(H[V_e])$ , because all vertices of  $e$  are contained in  $V_e$ .  $\lambda(H[V_e]) \geq d_e$  because every cut in  $H[V_e]$  has to cross some  $\pi(e')$  where  $e' \in T_e$ , and the weight of  $\pi(e')$  is at least  $d_e$ . Hence  $\gamma(e) \geq d_e$ .

We claim if we remove all edges in  $H$  with weight at most  $d_e$ , then it will disconnect some  $s, t \in e$ . If the claim is true then  $e$  crosses a cut of value at most  $p d_e$ . By [Lemma 2.2](#),  $\gamma(e) \leq p d_e$ . Assume that the claim is not true. Then, in the graph  $A(H)$  we can remove all edges with weight at most  $d_e$  and the vertices in  $e$  will still be connected. We can assume without loss of generality that the maximum weight spanning tree  $T$  in  $A(H)$  is computed using the greedy Kruskal's algorithm. This implies that  $T_e$  will contain only edges with weight strictly greater than  $d_e$ . This contradicts the definition of  $d_e$ .

$A(H)$  can be constructed in  $O(p)$  time. The maximum spanning tree  $T$  can be found in  $O(p + n \log n)$  time. We can construct a data structure on  $T$  in  $O(n)$  time, such that for any  $u, v \in V$ , it returns  $d_{uv}$  in  $O(1)$  time. [3] To compute  $d_e$ , we fix some vertex  $v$  in  $e$ , and compute  $d_e = \min_{u \in e, v \neq u} d_{uv}$  using the data structure in  $O(|e|)$  time. Computing  $d_e$  for all  $e$  takes in  $O(\sum_{e \in E} |e|) = O(p)$  time. The total running time is  $O(p + n \log n)$ . We conclude the following theorem.

**Lemma 4.2** *Given a weighted hypergraph  $H$ , we can find a value  $d_e$  for each edge  $e$ , such that  $d_e \leq \gamma(e) \leq p d_e$  in  $O(p + n \log n)$  time.*

The preceding lemma gives us the desired intervals. Using [Theorem 4.1](#), we have the desired theorem.

**Theorem 4.3** Given a rank  $r$  weighted hypergraph  $H$  with weight function  $w$ , in  $O(p \log^2 n \log p)$  time, one can find a  $O(r)$ -approximate strength function of  $H$ .

## References

- [1] Joshua Batson, Daniel A Spielman, and Nikhil Srivastava. Twice-ramanujan sparsifiers. *SIAM Journal on Computing*, 41(6):1704–1721, 2012.
- [2] András A. Benczúr and David R. Karger. Randomized approximation schemes for cuts and flows in capacitated graphs. *SIAM J. Comput.*, 44(2):290–319, 2015. Preliminary versions appeared in STOC ’96 and SODA ’98.
- [3] Bernard Chazelle. Computing on a free tree via complexity-preserving mappings. *Algorithmica*, 2(1):337–361, 1987.
- [4] Chandra Chekuri and Chao Xu. Computing minimum cuts in hypergraphs. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1085–1100, 2017.
- [5] Wai Shing Fung, Ramesh Hariharan, Nicholas J.A. Harvey, and Debmalya Panigrahi. A general framework for graph sparsification. In *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing*, STOC ’11, pages 71–80, New York, NY, USA, 2011. ACM.
- [6] Regina Klimmek and Frank Wagner. A simple hypergraph min cut algorithm. Technical Report B 96-02, Bericht FU Berlin Fachbereich Mathematik und Informatik, 1996. Available at [http://edocs.fu-berlin.de/docs/servlets/MCRFileNodeServlet/FUDOCs\\_derivate\\_000000000297/1996\\_02.pdf](http://edocs.fu-berlin.de/docs/servlets/MCRFileNodeServlet/FUDOCs_derivate_000000000297/1996_02.pdf).
- [7] Dmitry Kogan and Robert Krauthgamer. Sketching cuts in graphs and hypergraphs. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, ITCS ’15, pages 367–376, New York, NY, USA, 2015. ACM.
- [8] E. L. Lawler. Cutsets and partitions of hypergraphs. *Networks*, 3(3):275–285, 1973.
- [9] Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in  $\tilde{O}(\sqrt{\text{rank}})$  iterations and faster algorithms for maximum flow. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 424–433. IEEE, 2014.
- [10] Yin Tat Lee and He Sun. An SDP-Based Algorithm for Linear-Sized Spectral Sparsification. *ArXiv e-prints*, February 2017.
- [11] Wai-Kei Mak and D.F. Wong. A fast hypergraph min-cut algorithm for circuit partitioning. *Integration, the VLSI Journal*, 30(1):1 – 11, 2000.
- [12] Hiroshi Nagamochi and Toshihide Ibaraki. A linear-time algorithm for finding a sparse  $k$ -connected spanning subgraph of a  $k$ -connected graph. *Algorithmica*, 7(1-6):583–596, 1992.

## A $k$ -sparse certificate

We show how to find a  $k$ -sparse certificate for weighted and unweighted hypergraphs.

We refer to [4] for terminology. Given a hypergraph  $H = (V, E)$  consider an MA-ordering of vertices  $v_1, \dots, v_n$ . Let  $e_1, \dots, e_m$  be the induced head ordering of the edges.

For the unweighted version, let  $D_k(v)$  be the first  $k$  backward edges of  $v$  in the head ordering, or all the backward edges of  $v$  if there are fewer than  $k$  backward edges. Given  $H$  and  $k$ ,  $H_k = (V, E_k)$  is defined such that  $E_k = \bigcup_v D_k(v)$ . It is easy to see  $H_k$  can be constructed in linear time once the

MA-ordering have been computed. It is a  $k$ -sparse certificate follows directly from [4]. It is also obvious it contain at most  $k(n - 1)$  edges, since  $|D_k(v)| \leq k$  and  $D_k(v_1) = \emptyset$ .

The algorithm is similar for the weighted version. For a vertex  $v$ , assume its backward edges are  $e_1, \dots, e_\ell$  with weights  $w_1, \dots, w_\ell$ , and if  $i < j$  then  $e_i$  comes before  $e_j$  in the head order of the edges. Define  $w_v(e_i) = \max(\min(k - \sum_{j < i} w_j, w_i), 0)$  and 0 for all other edges. Define  $w'(e) = \max_{v \in V} w_v(e)$ . A weighted hypergraph with the weight function  $w'$  is a  $k$ -sparse certificate of  $H$ . With careful book-keeping,  $w'$  can be computed in  $O(p)$  time.

The running time is dominated by finding the MA-ordering, which is  $O(p + n \log n)$  time.